Journal of Computational and Applied Mathematics 330 (2018) 193–213

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

On POD-based Deflation Vectors for DPCG applied to porous media problems

G.B. Diaz Cortes [a,*], C. Vuik [a], J.D. Jansen [b]

[a] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

[b] Faculty of Civil Engineering and Geosciences, Delft University of Technology, Stevinweg 1, 2628 CN Delft, The Netherlands

ARTICLE INFO

Article history:
Received 6 February 2017

Keywords:
Deflation
POD
PCG
Single-phase flow
Heterogeneous porous media

ABSTRACT

We study fast and robust iterative solvers for large systems of linear equations resulting from simulation of flow through strongly heterogeneous porous media. We propose the use of preconditioning and deflation techniques, based on information obtained from the system, to reduce the time spent in the solution of the linear system.

An important question when using deflation techniques is how to find good deflation vectors, which lead to a decrease in the number of iterations and a small increase in the required computing time per iteration. In this paper, we propose the use of deflation vectors based on a POD-reduced set of snapshots. We investigate convergence and the properties of the resulting methods. Finally, we illustrate these theoretical results with numerical experiments. We consider compressible and incompressible single-phase flow in a layered model with variations in the permeability layers up to 10^3 and the SPE 10 benchmark model with a contrast in permeability coefficients of 10^7. Using deflation for the incompressible problem, we reduce the number of iterations to 1 or 2 iterations. With deflation, for the compressible problem, we reduce up to $\sim$ 80% the number of iterations when compared with the only-preconditioned solver.

1. Introduction

Often, most computational time spent during the simulation of multi-phase flow through porous media is taken up by the solution of the pressure equation. This involves, primarily, solving large systems of linear equations as part of the iterative solution of the time and space discretized, governing nonlinear partial differential equations. The time spent in solving the linear systems depends on the size of the problem and the heterogeneity, i.e. the spatial variations of rock permeability values within the medium (permeability is an inverse measure of the resistance to flow, which is related to the porosity and the pore structure of the rock). Solution of problems with extreme contrasts in the permeability values may lead to very large computing times.

Iterative methods are known to be the best option to solve such extreme problems. However, sometimes, iterative methods are not sufficient to solve these problems in a reasonable amount of time. As the systems become larger or ill-conditioned, finding a way to accelerate the convergence of these methods becomes necessary. Preconditioning is a way to accelerate convergence, but new preconditioning techniques still need to be developed to improve the performance of iterative methods [1,2]. Reduced Order Models (ROM) have also been studied to improve computational efficiency by

* Corresponding author.
E-mail address: g.b.diazcortes@tudelft.nl (G.B. Diaz Cortes).

http://dx.doi.org/10.1016/j.cam.2017.06.032
0377-0427/

reducing the model size without losing essential information [3–5]. A potential ROM to reduce the computing time for large-scale problems is Proper Orthogonal Decomposition (POD). A method that has been investigated for flow problems in porous media, in [6–15], among others. The use of a POD-based preconditioner for acceleration of the solution is proposed by Astrid et al. [11] to solve the pressure equation resulting from two-phase reservoir simulation, by Jiang et al. [14] for a similar application and by Pasetto et al. [15] for groundwater flow models.

The POD method requires the computation of a series of 'snapshots' which are solutions of the problem with slightly different parameters or well inputs. Astrid et al. [11] use snapshots in the form of solutions of the pressure equation computed in a small number of short pre-simulations, prior to the actual simulation. The snapshots are obtained with diverse well configurations. They report promising speed ups with factors between three and five. They note that the overhead required to pre-compute the POD solutions implies that the method will be particularly attractive when many solutions of near-similar simulation models are required. A similar approach is followed by Jiang [14], who concludes that POD-based pressure preconditioning does not appear to be an ideal choice, due to its dependence on the differences between the right-hand sides (forcing terms) used in the pre-simulations and the actual simulation. The snapshots computed by Pasetto et al. [15] are solutions of the previous time steps in the full-model. Once the snapshots are computed, the POD method is used to obtain a set of basis vectors that capture the most relevant features of the system, which can be used to speed-up the subsequent simulations.

The method of Pasetto et al. [15] is partly based on the work of Markovinovic and Jansen [8], who use a similar, but more restricted approach, in which the acceleration is achieved by only improving the initial guess.

Problems with high contrast between the permeability coefficients are, sometimes, approached through the use of deflation techniques, see, e.g., [16]. These techniques involve the search of good deflation vectors, which are, usually, problem-dependent. In [16], subdomain based deflation vectors are used for layered problems with a large contrast between permeability coefficients. However, these deflation vectors cannot be used if the distribution of the permeability coefficients is not structured, as is usually the case in reservoir simulation models; see, e.g., the well-known SPE 10 benchmark problem [17].

Algebraic Multigrid (AMG) [18], Multi-level and Domain Decomposition [19] preconditioners have been studied in combination deflation techniques to accelerate the convergence of iterative methods. In [8,11,15], after computing a basis from the previously obtained snapshots, the solution is computed in the subspace generated by this basis and then projected back to the original high-dimensional system. Carlberg et al. [20] also use POD to obtain information from the system, in particular, the previous time step solutions. Then, a Krylov-subspace is constructed using the information obtained previously.

Following the ideas of [8,11,15,20], we propose the use of POD of many snapshots to capture the system's behavior and combine this technique with deflation to accelerate the convergence of an iterative Krylov method. In this work, instead of computing the solution in a low dimensional subspace, the basis obtained with POD is proposed as an alternative choice for the deflation vectors to accelerate the convergence of the pressure solution in reservoir simulation.

This work is divided into six sections. Section 2 is devoted to a detailed description of the models used to simulate flow through a porous medium. In Section 3, we present some theory about the linear solvers used in this work and we introduce preconditioning and deflation techniques. In Section 4, we present some theory about POD. We prove two lemmas that will help us in the choice of good deflation vectors for the incompressible case studied in Section 5.

In Section 6, we present numerical experiments, we describe the studied problem, the solver and the preconditioning and deflation techniques used to speed up the solver. The results are also presented in this section. Finally, we end with the conclusions.

2. Flow through porous media

Petroleum reservoirs are layers of sedimentary rock, which vary in terms of their grain size and mineral contents. The volume fraction of the rock in-between the grains, i.e. the void space, is called *rock porosity*, a scalar quantity indicated with ϕ. The ability of the rock to transmit a single fluid, when the void space is completely filled with fluid, is known as *rock permeability*, a tensor quantity indicated with K.

Reservoir simulation is a way to analyze and predict the fluid behavior in a reservoir. The description of subsurface flow simulation involves two types of models: geological (static) and flow (dynamic) models. The static model is used to describe spatial properties of the reservoir, i.e. the porosities and permeabilities, which are parameters for the dynamic model. The dynamic model is subsequently used to predict fluid pressures and flow taking into account mass conservation and Darcy's law, an empirical, simplified version of the momentum conservation equations. The corresponding equations used to describe single-phase flow through a porous medium are (see, e.g. [21–23]):

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla\cdot(\rho v) = q, \qquad v = -\frac{K}{\mu}(\nabla p - \rho g \nabla z), \tag{1}$$

or

$$\frac{\partial(\rho\phi)}{\partial t} - \nabla\cdot\left(\frac{\rho K}{\mu}(\nabla p - \rho g \nabla z)\right) = q, \tag{2}$$

where the pressure p is the primary unknown, g is the constant of gravity, d is the reservoir depth, ρ and μ are the fluid density and viscosity and q is a source term (i.e. an injection or production well). A list of notation is presented in Appendix A. The fluid density $\rho = \rho(p)$ and the rock porosity $\phi = \phi(p)$ can be pressure-dependent. Rock porosity is related to the pressure via the rock compressibility. The relation is given by:

$$c_r = \frac{1}{\phi}\frac{d\phi}{dp} = \frac{d(ln(\phi))}{dp}.$$

If the rock compressibility is constant, the previous equation can be integrated as:

$$\phi(p) = \phi_0 e^{c_r(p-p_0)}. \tag{3}$$

The fluid density and the pressure are related via the fluid compressibility c_f, according to:

$$c_f = \frac{1}{\rho}\frac{d\rho}{dp} = \frac{d(ln(\rho))}{dp}.$$

If the fluid compressibility is constant, the previous equation can be integrated as:

$$\rho(p) = \rho_0 e^{c_f(p-p_0)}. \tag{4}$$

To solve Eq. (2), it is necessary to supply conditions at the boundary of the domain. While, for parabolic equations, we also need to impose initial conditions. Boundary and initial conditions will be discussed later for each problem.

2.1. Incompressible fluid

If the density and the porosity are not pressure-dependent in Eq. (2), we have an incompressible model, where density and porosity do not change over time. Therefore, the incompressible model is time-independent. Assuming no gravity terms and a fluid with constant viscosity, Eq. (2) then becomes:

$$-\frac{\rho}{\mu}\nabla\cdot(K\nabla p) = q. \tag{5}$$

Discretization

The spatial derivatives are approximated using a finite difference scheme with cell central differences. For a 3D model, we suppose a mesh with a uniform grid size Δx, Δy, Δz, where (i, j, l) is the center of the cell in the position i in the x direction, j in the y direction, and l in the z direction (x_i, y_j, z_l), and $p_{i,j,l} = p(x_i, y_j, z_l)$ is the pressure at this point.

For the x direction, we have (see, e.g. [21–23]):

$$\frac{\partial}{\partial x}\left(k\frac{\partial p}{\partial x}\right) = \frac{\Delta}{\Delta x}\left(k\frac{\Delta p}{\Delta x}\right) + \mathcal{O}(\Delta x^2) = \frac{k_{i+\frac{1}{2},j,l}(p_{i+1,j,l} - p_{i,j,l}) - k_{i-\frac{1}{2},j,l}(p_{i,j,l} - p_{i-1,j,l})}{(\Delta x)^2} + \mathcal{O}(\Delta x^2),$$

where $k_{i-\frac{1}{2},j,l}$ is the harmonic average of the permeability for cells $(i-1, j, l)$ and (i, j, l):

$$k_{i-\frac{1}{2},j,l} = \frac{2}{\frac{1}{k_{i-1,j,l}} + \frac{1}{k_{i,j,l}}}. \tag{6}$$

After discretization, Eq. (5), together with boundary conditions, can be written as:

$$\mathbf{Tp} = \mathbf{q}, \tag{7}$$

where $\mathbf{T}$ is known as the transmissibility matrix with elements in adjacent grid cells. The *transmissibility* ($T_{i-\frac{1}{2},j,l}$) between grid cells $(i-1, j, l)$ and (i, j, l) is defined as:

$$T_{i-\frac{1}{2},j,l} = \frac{2\Delta y\Delta z}{\mu\Delta x}k_{i-\frac{1}{2},j,l}. \tag{8}$$

System (7) is a linear system, that can be solved with iterative or direct methods. For the solution of this system, it is necessary to define boundary conditions in all boundaries of the domain. These conditions can be prescribed pressures (Dirichlet conditions), flow rates (Neumann conditions) or a combination of these (Robin conditions).

2.2. Compressible fluid

If the fluid is compressible, with a constant compressibility, the density depends on the pressure (see Eq. (4)). Therefore, Eqs. (1) become:

$$\frac{\partial(\rho(p)\phi)}{\partial t} + \nabla\cdot(\rho(p)v) = q, \qquad v = -\frac{K}{\mu}(\nabla p - \rho(p)g\nabla z). \tag{9}$$

Discretization

Using backward Euler time discretization, Eqs. (9) are approximated by:

$$\frac{(\phi\rho(p))^{n+1} - (\phi\rho(p))^n}{\Delta t^n} + \nabla\cdot(\rho(p)v)^{n+1} = q^{n+1},$$

$$v^{n+1} = -\frac{K}{\mu^{n+1}}(\nabla(p^{n+1}) - g\rho^{n+1}\nabla z). \tag{10}$$

Assuming no gravity terms, constant fluid viscosity and constant rock porosity, Eqs. (10) become:

$$\phi\frac{\rho(p^{n+1}) - \rho(p^n)}{\Delta t^n} - \frac{1}{\mu}\nabla\cdot(\rho(p^{n+1})K\nabla p^{n+1}) + q^{n+1} = 0. \tag{11}$$

Due to the dependence of ρ on the pressure, the latter is a nonlinear equation for p, that can be linearized with, e.g., the Newton–Raphson (NR) method. Eq. (11) can be discretized in space, using, e.g. a finite differences scheme. After spatial discretization, Eq. (11) reads:

$$\frac{\mathbf{V}(\mathbf{p}^{n+1}) - \mathbf{V}(\mathbf{p}^n)}{\Delta t^n} + \mathbf{T}\mathbf{p}^{n+1} = \mathbf{q}^{n+1}. \tag{12}$$

We note that in a more general case, where also the porosity is pressure-dependent, a slightly more complex, mass conservative formulation is usually employed (see [21–23]). As in the incompressible case, we need to define boundary condition to solve Eq. (12). Dirichlet, Neumann or Robin boundary conditions can be used. For this problem, we also have a derivative with respect to time. Therefore, it is also necessary to specify the initial conditions that are the pressure values of the reservoir at the beginning of the simulation.

2.3. Well model

In reservoirs, wells are typically drilled to extract or inject fluids. Fluids are injected into a well or produced from a well at constant rate or constant bottom-hole pressure (bhp).

When the bhp is prescribed, the flow rates into or from the wells are usually computed with the aid of a well model, that takes into account the bhp and the average grid pressure in the block containing the well. This model is a linear relationship between the bhp and the flow rate in a well. For a cell (i, j, l), that contains a well, this relationship is given by:

$$q_{(i,j,l)} = I_{(i,j,l)}(p_{(i,j,l)} - p_{bh(i,j,l)}), \tag{13}$$

where $I_{(i,j,l)}$ is the productivity or injectivity index of the well, $p_{(i,j,l)}$ is the reservoir pressure in the cell where the well is located, and $p_{bh(i,j,l)}$ is a prescribed pressure inside the well.

Incompressible fluid

Using the well model for an incompressible fluid, Eq. (7) transforms into:

$$\mathbf{T}\mathbf{p} = \mathbf{I}_w(\mathbf{p} - \mathbf{p}_{bh}), \tag{14}$$

where $\mathbf{I}_w$ is a diagonal matrix containing the productivity or injectivity indices of the wells present in the reservoir. The diagonal elements are zero for cells without wells and have the value of the well index for each cell containing a well.

Compressible fluid

For a compressible fluid, using the well model, Eq. (12) reads:

$$\phi\frac{\rho(\mathbf{p}^{n+1}) - \rho(\mathbf{p}^n)}{\Delta t^n} - \frac{1}{\mu}\nabla\cdot(\rho(\mathbf{p}^{n+1})\mathbf{K}\nabla\mathbf{p}^{n+1}) + \mathbf{I}_w(\mathbf{p}^{n+1} - \mathbf{p}_{bh}^{n+1}) = \mathbf{0}, \tag{15}$$

or

$$\frac{\mathbf{V}(\mathbf{p}^{n+1}) - \mathbf{V}(\mathbf{p}^n)}{\Delta t^n} + (\mathbf{T} + \mathbf{I}_w)\mathbf{p}^{n+1} - \mathbf{I}_w(\mathbf{p}_{bh}^{n+1}) = \mathbf{0}.$$

2.4. Solution procedure for compressible flow

As mentioned before, for the compressible problem, we have a nonlinear system, that depends on the pressure at the time step n and the pressure at time step $n + 1$:

$$\mathbf{g}(\mathbf{p}^{n+1}; \mathbf{p}^n) = 0. \tag{16}$$

This nonlinear system can be solved with the NR method. The resulting system for the $(k + 1)$th NR iteration is:

$$\mathbf{J}(\mathbf{p}^k)\delta\mathbf{p}^{k+1} = -\mathbf{g}(\mathbf{p}^k; \mathbf{p}^n), \qquad \mathbf{p}^{k+1} = \mathbf{p}^k + \delta\mathbf{p}^{k+1},$$

where $\mathbf{J}(\mathbf{p}^k) = \frac{\partial \mathbf{g}(\mathbf{p}^k;\mathbf{p}^n)}{\partial \mathbf{p}^k}$ is the Jacobian matrix, and $\delta\mathbf{p}^{k+1}$ is the NR update at iteration step $k+1$.

Therefore, the linear system to solve is:

$$\mathbf{J}(\mathbf{p}^k)\delta\mathbf{p}^{k+1} = \mathbf{b}(\mathbf{p}^k) \tag{17}$$

with $\mathbf{b}(\mathbf{p}^k)$ being the function evaluated at iteration step k, $\mathbf{b}(\mathbf{p}^k) = -\mathbf{g}(\mathbf{p}^k; \mathbf{p}^n)$.

The procedure to solve a compressible flow problem consists of three stages. During the first stage, we increase the time with one time step and solve Eq. (15) for the new time. Because of the nonlinearity of Eq. (15), we use an iterative Newton–Raphson procedure that involves linearization at each iteration, i.e. we perform a series of iterations to find the zeros of Eq. (16). For every NR iteration, the linear system in Eq. (17) is solved. In this work, the solution of the linear system is performed with iterative methods (see Section 3). A summary of this procedure is presented in Algorithm 1.

Algorithm 1

```
for t = 0, ...,                                          %Time integration
    Select time step
    for NR_iter = 0, ...,                                %NR iteration
        Find zeros of g(p^{n+1}; p^n) = 0
        for lin_iter = 0, ...,                           %Linear iteration
            Solve J(p^k)δp^{k+1} = b(p^k) for each NR iteration
        end
    end
end
```

3. Iterative solution methods

When simulating single-phase flow through a porous medium, we obtain a linear system of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{18}$$

for both compressible and incompressible models. Since $\mathbf{A}$ is SPD, we choose Conjugate Gradient (CG) as iterative method, accelerated with the Incomplete Cholesky (IC) preconditioner. In this work, we also study the acceleration with deflation techniques. In this section, we give a brief overview of the methods.

3.1. Conjugate gradient method

Given a starting solution $\mathbf{x}^0$ and the residual defined by $\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$, we define the Krylov subspace $\mathcal{K}_k$ as $\mathcal{K}_k(\mathbf{A}, \mathbf{r}^0) = span\{\mathbf{r}^0, \mathbf{A}\mathbf{r}^0, \ldots, \mathbf{A}^{k-1}\mathbf{r}^0\}$ and $\mathbf{x}^k \in \mathbf{x}^0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}^0)$ has a minimal error measured in the $\mathbf{A}$-norm, for all the approximations contained in $\mathbf{x}^0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}^0)$. The error of this approximation is bounded by:

$$\|\mathbf{x} - \mathbf{x}^{k+1}\|_{\mathbf{A}} \leq 2\|\mathbf{x} - \mathbf{x}^0\|_{\mathbf{A}} \left(\frac{\sqrt{\kappa_2(\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{A})} + 1} \right)^{k+1}.^{1} \tag{19}$$

The pseudo code for CG is given in Algorithm 2.

Algorithm 2 Conjugate Gradient (CG) method, solving $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Give an initial guess $\mathbf{x}^0$.
Compute $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$ and set $\mathbf{p}^0 = \mathbf{r}^0$.
for $k = 0, \ldots,$ until convergence
$\alpha^k = \frac{(\mathbf{r}^k, \mathbf{r}^k)}{(\mathbf{A}\mathbf{p}^k, \mathbf{p}^k)}$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
$\beta^k = \frac{(\mathbf{r}^{k+1}, \mathbf{r}^{k+1})}{(\mathbf{r}^k, \mathbf{r}^k)}$
$\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta^k \mathbf{p}^k$
end

[1] The condition number $\kappa_2(\mathbf{A})$ is defined as $\kappa_2(\mathbf{A}) = \frac{\sqrt{\lambda_{max}(\mathbf{A}^T\mathbf{A})}}{\sqrt{\lambda_{min}(\mathbf{A}^T\mathbf{A})}}$. If $\mathbf{A}$ is SPD, $\kappa_2(\mathbf{A}) = \frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})}$ [24].

3.2. Preconditioning

To accelerate the convergence of a Krylov method, one can transform the system into another one, containing an iteration matrix with a better spectrum, i.e, a smaller condition number. This can be done by multiplying the system from Eq. (18) by a matrix $\mathbf{M}^{-1}$.

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}. \tag{20}$$

The new system has the same solution, but can provide a substantial reduction of the condition number. For this preconditioned system, the error is bounded by:

$$\|\mathbf{x} - \mathbf{x}^k\|_{\mathbf{A}} \leq 2\|\mathbf{x} - \mathbf{x}^0\|_{\mathbf{A}} \left(\frac{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{A})} + 1} \right)^k. \tag{21}$$

The matrix $\mathbf{M}$ is chosen as an *SPD* matrix such that $\kappa(\mathbf{M}^{-1}\mathbf{A}) \leq \kappa(\mathbf{A})$, and $\mathbf{M}^{-1}\mathbf{b}$ is cheap to compute.

3.3. Deflation

Deflation is used to annihilate the effect of extreme eigenvalues on the convergence of an iterative method [16]. Given an *SPD* matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, for a given matrix $\mathbf{Z} \in \mathbb{R}^{n\times m}$ the deflation matrix $\mathbf{P}$ is defined as follows [19,25]:

$$\mathbf{P} = \mathbf{I} - \mathbf{A}\mathbf{Q}, \qquad \mathbf{P} \in \mathbb{R}^{n\times n}, \qquad \mathbf{Q} \in \mathbb{R}^{n\times n}, \tag{22}$$

where

$$\mathbf{Q} = \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^T, \qquad \mathbf{Z} \in \mathbb{R}^{n\times m}, \qquad \mathbf{E} \in \mathbb{R}^{m\times m},$$

with

$$\mathbf{E} = \mathbf{Z}^T\mathbf{A}\mathbf{Z}.$$

The matrix $\mathbf{E}$ is known as the *Galerkin* or *coarse* matrix, which has to be invertible. If $\mathbf{A}$ is *SPD* and $\mathbf{Z}$ is full rank, then, $\mathbf{E}$ is invertible. The full rank matrix $\mathbf{Z}$ is called the *deflation − subspace* matrix, and its columns are the *deflation* vectors or *projection* vectors.

Deflated PCG method

To obtain the solution of linear system (18), we have to solve the deflated system (see Appendix D):

$$\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{P}\mathbf{b}, \tag{23}$$

with the CG method, for the deflated solution $\hat{\mathbf{x}}$. This deflated the solution is related to the solution $\mathbf{x}$ of the original system as (see Appendix D):

$$\mathbf{x} = \mathbf{Q}\mathbf{b} + \mathbf{P}^T\hat{\mathbf{x}}. \tag{24}$$

The deflated linear system can also be preconditioned with an *SPD* matrix $\mathbf{M}$. After preconditioning, the deflated preconditioned system to solve with CG is [19]:

$$\tilde{\mathbf{P}}\tilde{\mathbf{A}}\hat{\tilde{\mathbf{x}}} = \tilde{\mathbf{P}}\tilde{\mathbf{b}},$$

where:

$$\tilde{\mathbf{A}} = \mathbf{M}^{-\frac{1}{2}}\mathbf{A}\mathbf{M}^{-\frac{1}{2}}, \qquad \hat{\tilde{\mathbf{x}}} = \mathbf{M}^{\frac{1}{2}}\hat{\mathbf{x}}, \qquad \tilde{\mathbf{b}} = \mathbf{M}^{-\frac{1}{2}}\mathbf{b}.$$

This method is called the Deflated Preconditioned Conjugate Gradient *DPCG* method. In practice $\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{P}\mathbf{b}$ is computed and the error is bounded by:

$$\|\mathbf{x} - \mathbf{x}^{i+1}\|_{\mathbf{A}} \leq 2\|\mathbf{x} - \mathbf{x}^0\|_{\mathbf{A}} \left(\frac{\sqrt{\kappa_{eff}(\mathbf{M}^{-1}\mathbf{P}\mathbf{A})} - 1}{\sqrt{\kappa_{eff}(\mathbf{M}^{-1}\mathbf{P}\mathbf{A})} + 1} \right)^{i+1},$$

were $\kappa_{eff} = \frac{\lambda_{max}(M^{-1}PA)}{\lambda_{min}(M^{-1}PA)}$ is the effective condition number and $\lambda_{min}(M^{-1}PA)$ is the smallest non-zero eigenvalue of $M^{-1}PA$.

Choices of deflation vectors

The deflation method is used to remove the effect of the most unfavorable eigenvalues of $\mathbf{A}$. If the matrix $\mathbf{Z}$ contains eigenvectors corresponding to the unfavorable eigenvalues, the convergence of the iterative method is achieved faster. However, to obtain and to apply the eigenvectors is costly in view of memory and CPU time. Therefore, a good choice of the matrix $\mathbf{Z}$ that efficiently approximates the eigenvectors is essential for the applicability of the method.

A good choice of the deflation vectors is usually problem-dependent. Available information on the system is, in general, used to obtain these vectors. Most of the techniques used to choose deflation vectors are based on approximating eigenvectors, recycling [26], subdomain deflation vectors [1] or multigrid and multilevel based deflation techniques [19,27]. A summary of these techniques is given below.

Recycling deflation. A set of search vectors previously used is reused to build the deflation-subspace matrix [26]. The vectors could be, for example, $q-1$ solution vectors of the linear system with different right-hand sides or of different time steps. The matrix $\mathbf{Z}$ containing these solutions is:

$$\mathbf{Z} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(q-1)}].$$

Subdomain deflation. The domain is divided into several subdomains, using domain decomposition techniques or taking into account the properties of the problem. For each subdomain, there is a deflation vector that contains ones for cells in the subdomain and zeros for cells outside [1].

Multigrid and multilevel deflation. For the multigrid and multilevel methods, the prolongation and restriction matrices are used to pass from one level or grid to another. These matrices can be used as the deflation-subspace matrices $\mathbf{Z}$ [19].

4. Proper Orthogonal Decomposition (POD)

As mentioned before, in this work, we want to combine deflation techniques and Proper Orthogonal Decomposition (POD) to reduce the number of iterations necessary to solve the linear system obtained from reservoir simulation, in a cheap and, automatic way. In this section, we give a brief overview of the POD method.

The POD method is a Model Order Reduction (MOR) method, where a high-order model is projected onto a space spanned by a small set of orthonormal basis vectors. The high dimensional variable $\mathbf{x} \in \mathbb{R}^n$ is approximated by a linear combination of l orthonormal basis vectors [11]:

$$\mathbf{x} \approx \sum_{i=1}^{l} c_i \psi_i, \tag{25}$$

where, $\psi_i \in \mathbb{R}^n$ are the basis vectors and c_i are their corresponding coefficients. In matrix notation, Eq. (25) is rewritten as:

$$\mathbf{x} \approx \Psi \mathbf{c},$$

where $\Psi = [\psi_1\ \psi_2\ \ldots\ \psi_l]$, $\Psi \in \mathbb{R}^{n\times l}$ is the matrix containing the basis vectors, and $\mathbf{c} \in \mathbb{R}^l$ is the vector containing the coefficients of the basis vectors.

The basis vectors ψ_i are computed from a set of 'snapshots' $\{\mathbf{x_i}\}_{i=1,\ldots,m}$, obtained by simulation or experiments [8]. In POD, the basis vectors $\{\psi_j\}_{j=1}^{l}$, are l eigenvectors, corresponding to the largest eigenvalues, $\{\sigma_j\}_{j=1}^{l}$, of the data snapshot correlation matrix $\mathbf{R}$.

$$\mathbf{R} := \frac{1}{m}\mathbf{X}\mathbf{X}^T \equiv \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^T, \qquad \mathbf{X} := [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_m], \tag{26}$$

where $\mathbf{X} \in \mathbb{R}^{n\times m}$ is an SPSD matrix containing the previously obtained snapshots. The l eigenvectors should contain almost all the variability of the snapshots. Usually, they are chosen as the eigenvectors of the maximal number (l) of eigenvalues satisfying [8]:

$$\frac{\sum_{j=1}^{l} \sigma_j}{\sum_{j=1}^{m} \sigma_j} \leq \alpha, \qquad 0 < \alpha \leq 1, \tag{27}$$

with α close to 1. The eigenvalues σ_j are ordered from large to small with σ_1 the largest eigenvalue of $\mathbf{R}$. It is not necessary to compute the eigenvalues from $\mathbf{X}\mathbf{X}^T$, but instead, it is possible to compute the eigenvalues of the much smaller matrix $\mathbf{X}^T\mathbf{X}$ (see Appendix C).

In this study, we normalize the snapshots, so $\|\mathbf{x}_i\|_2 = 1$.

5. Deflation vector analysis

As mentioned in Section 3, it is important to choose 'good' deflation vectors if we want to speed up an iterative method.

We can use solutions of systems slightly different from the original (snapshots) as deflation vectors. For this, we need to choose a way of selecting these snapshots. The idea behind this selection is to obtain a small number of snapshots and, at the same time, obtain the largest amount of information from the system.

In this section, two lemmas are proved. The lemmas are helpful to select the systems used to obtain the snapshots.

Lemma 1. *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *be a non-singular matrix, and* $\mathbf{x}$ *be the solution of:*

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{28}$$

Let $\mathbf{x}_i, \mathbf{b}_i \in \mathbb{R}^n$, $i = 1, \ldots, m$, *be vectors linearly independent (l.i.) and*

$$\mathbf{A}\mathbf{x}_i = \mathbf{b}_i. \tag{29}$$

The following equivalence holds

$$\mathbf{x} = \sum_{i=1}^{m} c_i\mathbf{x}_i \qquad \Leftrightarrow \qquad \mathbf{b} = \sum_{i=1}^{m} c_i\mathbf{b}_i. \tag{30}$$

Proof ($\Rightarrow$)**.**

$$\mathbf{x} = \sum_{i=1}^{m} c_i\mathbf{x}_i \Rightarrow \mathbf{b} = \sum_{i=1}^{m} c_i\mathbf{b}_i. \tag{31}$$

Substituting $\mathbf{x}$ from (31) into $\mathbf{A}\mathbf{x} = \mathbf{b}$ leads to:

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^{m} \mathbf{A}c_i\mathbf{x}_i = \mathbf{A}(c_1\mathbf{x}_1 + \cdots + c_m\mathbf{x}_m).$$

Using the linearity of $\mathbf{A}$ the equation above can be rewritten as:

$$\mathbf{A}c_1\mathbf{x}_1 + \cdots + \mathbf{A}c_m\mathbf{x}_m = c_1\mathbf{b}_1 + \cdots + c_m\mathbf{b}_m = \mathbf{B}\mathbf{c} \tag{32}$$

where $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{c} \in \mathbb{R}^m$, and the columns of $\mathbf{B}$ are the vectors $\mathbf{b}_i$.

From (28) and (32) we get:

$$\mathbf{A}\mathbf{x} = \mathbf{b} = c_1\mathbf{b}_1 + \cdots + c_m\mathbf{b}_m = \sum_{i=1}^{m} c_i\mathbf{b}_i.$$

Proof ($\Leftarrow$)**.**

$$\mathbf{x} = \sum_{i=1}^{m} c_i\mathbf{x}_i \Leftarrow \mathbf{b} = \sum_{i=1}^{m} c_i\mathbf{b}_i. \tag{33}$$

Substituting $\mathbf{b}$ from (33) into $\mathbf{A}\mathbf{x} = \mathbf{b}$ leads to:

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^{m} c_i\mathbf{b}_i. \tag{34}$$

Since $\mathbf{A}$ is non-singular, multiplying (29) and (33) by $\mathbf{A}^{-1}$ we obtain:

$$\mathbf{x}_i = \mathbf{A}^{-1}\mathbf{b}_i,$$

$$\mathbf{x} = \mathbf{A}^{-1}\sum_{i=1}^{m} c_i\mathbf{b}_i = \sum_{i=1}^{m} c_i\mathbf{A}^{-1}\mathbf{b}_i,$$

then

$$\mathbf{x} = \sum_{i=1}^{m} c_i\mathbf{x}_i. \quad \boxtimes \tag{35}$$

Lemma 2. *If the deflation matrix* $\mathbf{Z}$ *is constructed with a set of* m *vectors*

$$\mathbf{Z} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \cdots & \mathbf{x}_m \end{bmatrix}, \tag{36}$$

such that $\mathbf{x} = \sum_{i=1}^{m} c_i\mathbf{x}_i$, *with* $\mathbf{x}_i$ *l.i., then the solution of system* (28) *is obtained with one iteration of DCG.*

Proof. The relation between $\hat{\mathbf{x}}$ and $\mathbf{x}$ is given in Eq. (24):

$$\mathbf{x} = \mathbf{Q}\mathbf{b} + \mathbf{P}^T\hat{\mathbf{x}}.$$

For the first term $\mathbf{Qb}$, taking $\mathbf{b} = \sum_{i=1}^{m} c_i \mathbf{b}_i$ we have:

$$\begin{aligned}
\mathbf{Qb} &= \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^T \left(\sum_{i=1}^{m} c_i \mathbf{b}_i \right) \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T \left(\sum_{i=1}^{m} c_i \mathbf{A}\mathbf{x}_i \right) \qquad \text{using Lemma 1} \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T \left(\mathbf{A}\mathbf{x}_1 c_1 + \cdots + \mathbf{A}\mathbf{x}_m c_m \right) \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T (\mathbf{A}\mathbf{Z}\mathbf{c}) \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})\mathbf{c} \\
&= \mathbf{Z}\mathbf{c} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + c_3\mathbf{x}_3 + c_4\mathbf{x}_4 + c_5\mathbf{x}_5 \\
&= \sum_{i=1}^{m} c_i \mathbf{x}_i = \mathbf{x}.
\end{aligned}$$

Therefore,

$$\mathbf{x} = \mathbf{Qb}, \tag{37}$$

is the solution to the original system.

For the second term of Eq. (24), $\mathbf{P}^T\hat{\mathbf{x}}$, we compute $\hat{\mathbf{x}}$ from Eq. (23):

$$\begin{aligned}
\mathbf{PA}\hat{\mathbf{x}} &= \mathbf{Pb} \\
\mathbf{AP}^T\hat{\mathbf{x}} &= (\mathbf{I} - \mathbf{AQ})\mathbf{b} \qquad \text{using Appendix D(f) and definition of } \mathbf{P}, \\
\mathbf{AP}^T\hat{\mathbf{x}} &= \mathbf{b} - \mathbf{AQb} \\
\mathbf{AP}^T\hat{\mathbf{x}} &= \mathbf{b} - \mathbf{Ax} = 0 \qquad \text{taking } \mathbf{Qb} = \mathbf{x} \text{ from above}, \\
\mathbf{P}^T\hat{\mathbf{x}} &= 0 \qquad \text{as } \mathbf{A} \text{ is invertible.}
\end{aligned}$$

Then we have obtain the solution

$$\mathbf{x} = \mathbf{Qb} + \mathbf{P}^T\hat{\mathbf{x}} = \mathbf{Qb},$$

in one step of DCG. ⊠

5.1. Accuracy of the snapshots

If we use an iterative method to obtain an approximate solution $\mathbf{x}^k$ for the system $\mathbf{Ax} = \mathbf{b}$, we cannot compute the relative error e_r (Eq. (38)) of the approximation with respect to the true solution, because the true solution is unknown,

$$e_r = \frac{\|\mathbf{x} - \mathbf{x}^k\|_2}{\|\mathbf{x}\|_2}. \tag{38}$$

Instead, we compute the relative residual r_r (Eq. (39)),

$$r_r = \frac{\|\mathbf{r}^k\|_2}{\|\mathbf{b}\|_2} \leq \epsilon, \tag{39}$$

and we set a stopping criterion ϵ or tolerance, that is related to the relative error as follows [28] (see Appendix B),

$$\frac{\|\mathbf{x} - \mathbf{x}^k\|_2}{\|\mathbf{x}\|_2} \leq \kappa_2(\mathbf{A})\epsilon = r_r.$$

Various tolerance values can be used in the experiments for the snapshots, as well as, for the solution of the original system.

If the maximum relative residual for the snapshots ($\mathbf{x}_i$) is $\epsilon = 10^{-\eta}$, then, the error in the snapshots is given by

$$\frac{\|\mathbf{x}_i - \mathbf{x}_i^k\|_2}{\|\mathbf{x}_i\|_2} \leq \kappa_2(\mathbf{A}) \times 10^{-\eta} = r_r.$$

From Eq. (35), if we compute m snapshots with an iterative method such that the solution of $\mathbf{x}$ is a linear combination of these vectors, after one iteration of DCG we obtain

$$\mathbf{x}^1 = \sum_{i=1}^{m} c_i \mathbf{x}_i^{1(i)},$$

where $\mathbf{x}_i^{1(i)}$ is the approximated solution of the snapshot i after one DCG iteration.

The error of this solution is given by:

$$\frac{\|\mathbf{x}-\mathbf{x}^1\|_2}{\|\mathbf{x}\|_2} = \frac{\|\sum_{i=1}^m c_i(\mathbf{x}_i-\mathbf{x}_i^1)\|_2}{\|\sum_{i=1}^m c_i\mathbf{x}_i\|_2} \leq \frac{\sum_{i=1}^m |c_i| \times \kappa_2(\mathbf{A}) \times 10^{-\eta}}{\|\sum_{i=1}^m c_i\mathbf{x}_i\|_2},$$

which means that the approximation has an error of the order $\kappa_2(\mathbf{A}) \times 10^{-\eta}$.

From Lemma 2 we know that if we use the snapshots $\mathbf{x}_i$ as deflation vectors, for the deflation method the solution is given by (Eq. (37)):

$$\mathbf{x} = \mathbf{Q}\mathbf{b}.$$

If the approximation $\mathbf{x}^1$ has an error of the order $\kappa_2(\mathbf{A}) \times 10^{-\eta}$, then, the solution achieved with the deflation method will have the same error,

$$\mathbf{Q}\mathbf{b} - \mathbf{x}^1 = \kappa_2(\mathbf{A}) \times 10^{-\eta}.$$

Therefore, it is important to take into account the condition number of the matrix to estimate the accuracy of the deflation vectors.

5.2. Boundary conditions

From Lemma 2, we know that if we use as deflation vectors a set of m snapshots

$$\mathbf{Z} = [\mathbf{x}_1 \quad \dots \quad \mathbf{x}_m],$$

such that $\mathbf{x} = \sum_{i=1}^m c_i\mathbf{x}_i$, where $\mathbf{x}$ is the solution of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$, the solution of the latter system is achieved with one DCG iteration.

In our application, only a small number (m) of elements of the right-hand side vector $\mathbf{b}$ can be changed. This implies that every $\mathbf{b}$ can be written as $\mathbf{b} = \sum_{i=1}^m c_i\mathbf{b}_i$. Using Lemma 1, this implies that $\mathbf{x}$ is such that $\mathbf{x} \in span\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$, which is called the solution span. Therefore, it is necessary to find the solution span of the system, such that the sum of the elements in the solution span and the sum of right-hand sides give as result the original system. In this section, we explore the subsystems that should be chosen, depending on the boundary conditions of the original system.

Neumann boundary conditions

When we have Neumann boundary conditions everywhere, the resulting matrix $\mathbf{A}$ is singular, and $\mathbf{A}[1 \quad 1 \quad \dots \quad 1 \quad 1]^T = \mathbf{0}$, $Ker(\mathbf{A}) = span([1 \quad 1 \quad \dots \quad 1 \quad 1]^T)$. Note that $\mathbf{A}\mathbf{x} = \mathbf{b}$ has only a solution if $\mathbf{b} \in span\{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ (with $\mathbf{a}_i$ the ith column of $\mathbf{A}$), which is equivalent to $\mathbf{b} \perp Ker(\mathbf{A})$ [29]. This implies that, if we have m sources with value s_i for the vector $\mathbf{b}_i$, we need that

$$\sum_{j=1}^m s_i^j = 0.$$

Then, for each nonzero right-hand side, we need to have at least two sources. Therefore, we can have at most $m-1$ linearly independent right-hand sides $\mathbf{b}_i$ containing two sources.

This means that the solution space has dimension $m-1$ and it can be spanned by $span\{\mathbf{x}_1, \dots, \mathbf{x}_{m-1}\}$. Each of these subsystems will have the same no-flux conditions (Neumann) in all the boundaries. As the original system is a linear combination of the subsystems (Lemma 1), the deflation vectors can be chosen as the solutions corresponding to the subsystems. Therefore, the deflation matrix will be given by:

$$\mathbf{Z} = [\mathbf{x}_1 \quad \dots \quad \mathbf{x}_{m-1}],$$

and if the accuracy of the snapshots used as deflation vectors is good enough (see Section 5.1), the solution is expected to be achieved in one DCG iteration.

Dirichlet boundary conditions

In this case, the right-hand side of the system can contain the values of the boundary $\mathbf{b}_b$ and the sources of the system $\mathbf{s}_i$. If we have m sources, as in the previous case, the right-hand side will be given by:

$$\mathbf{b} = \sum_{i=1}^m c_i\mathbf{s}_i + \mathbf{b}_b.$$

The subsystems will be $m+1$, where one of them corresponds to the boundary conditions $\mathbf{A}\mathbf{x}_b = \mathbf{b}_b$, and the other m will correspond to the sources $\mathbf{A}\mathbf{x}_i = \mathbf{s}_i$. Therefore, snapshot $m+1$ will be the solution $\mathbf{x}_b$ of the system with no sources and the Dirichlet boundary conditions of the original system. The other m snapshots will correspond to the m sources with homogeneous Dirichlet boundary conditions. Then, the solution space will be given by $span\{\mathbf{x}_1, \dots, \mathbf{x}_m, \mathbf{x}_b\}$. If we use the solution of the $m+1$ snapshots as deflation vectors, with the correct accuracy, we will obtain the solution within one DCG iteration.

6. Numerical experiments

6.1. Model

We study the solution of systems of linear equations resulting from the discretization of elliptic and parabolic partial differential equations for the description of single-phase flow through a porous medium. The solution of the linear system is performed with the Conjugate Gradient method preconditioned with Incomplete Cholesky (ICCG) and the Deflated Conjugate Gradient method preconditioned with Incomplete Cholesky (DICCG). We propose the use of snapshots and snapshots-based POD basis vectors as deflation vectors for the above-mentioned method.

In the present section, we give a general overview of the experiments we perform, but the specifications are presented below for each problem separately. We solve the elliptic problem (incompressible flow) and the parabolic problem (compressible flow). Neumann homogeneous boundary conditions (no-flux) are imposed for an academic layered problem and for the SPE 10 benchmark problem.

The model

The experiments simulate flow through a porous medium with a constant porosity field of 0.2. We model incompressible and compressible single-phase flow. For the single-phase model the following properties of the fluid are used:

- $\mu = 1$ cp,
- $\rho = 1014$ kg/m^3.

In the compressible case, the compressibility of the fluid is:

- $c_f = 1 \times 10^{-3}$.

The matrices corresponding to the linear systems $\mathbf{A}$ and right-hand sides $\mathbf{b}$ are obtained with the Matlab Reservoir Simulation Toolbox (MRST) [30].

Snapshots

As mentioned before, for the DICCG method we need a set of deflation vectors. In the first series of experiments (incompressible model), the deflation vectors are solutions of the system with various wells configurations. These solutions, called *snapshots*, are obtained with the ICCG method. For the compressible problem, the snapshots are the solutions of the previous time steps with the same well configuration. We also propose the use of a POD basis as deflation vectors, obtained from the previously computed snapshots for the incompressible and compressible cases. As tolerance or stopping criterion, we use the relative residual, defined as the 2-norm of the residual of the k-th iteration divided by the 2-norm of the right-hand side of the preconditioned or deflated system:

$$r_r = \frac{\|\mathbf{M}^{-1}r^k\|_2}{\|\mathbf{M}^{-1}b\|_2} \leq \epsilon.$$

The residual of the approximation is the same for both methods, ICCG and DICCG (see Appendix D.1). Therefore, the accuracy of the solution is the same for both methods.

6.2. Incompressible problem

We simulate single-phase flow through a porous medium, for an incompressible fluid (see Eq. (7)), with the previously mentioned characteristics. Homogeneous Neumann boundary conditions are imposed on all boundaries. This model contains five wells, four on the corners and one in the center of the domain.

A set of four linearly independent snapshots is used as deflation vectors (DICCG$_4$). We also use a linearly dependent set of 15 snapshots (DICCG$_5$) and a 4-vectors POD basis obtained from these 15 snapshots (DICCG$_{POD_4}$). We set the same boundary conditions as in the original problem for all the snapshots. The four linearly independent snapshots ($\mathbf{z}_1$-$\mathbf{z}_4$) are obtained giving a value of zero to one well and non-zero values to the other wells, such that the sum of the well pressures equals zero. The set of 15 snapshots are all possible combinations of wells such that the flow-in equals the flow-out of the reservoir. The snapshots and the solutions are obtained with a tolerance of 10^{-11}.

A summary of the well configurations is presented in Table 1.

Heterogeneous permeability layers

A Cartesian grid of 64 $\times$ 64 grid cells and length (L_x, L_y) of 70 $\times$ 70 m^2 with 8 layers of the same size is studied. Four layers have a permeability σ_1 and each layer is followed by a other one with a different permeability value σ_2 (see Fig. 1). The permeability of one set of layers is set to $\sigma_1 = 1$ mD, the permeability of the other set σ_2 is varied. Therefore, the contrast in permeability between the layers ($\frac{\sigma_2}{\sigma_1} = \sigma_2$) depends on the value of σ_2.

We investigate the dependence on the contrast in permeability value between the layers for the ICCG and DICCG methods. The permeability σ_2 varies from $\sigma_2 = 10^{-1}$ mD to $\sigma_2 = 10^{-3}$ mD.

Table 1
Table with the well configurations of the system and the snapshots.

System configuration					
Well pressures (bars)					
	$W1$	$W2$	$W3$	$W4$	$W5$
	-1	-1	-1	-1	4
Snapshots (linearly independent)					
	$W1$	$W2$	$W3$	$W4$	$W5$
$\mathbf{z}_1$	0	-1	-1	-1	3
$\mathbf{z}_2$	-1	0	-1	-1	3
$\mathbf{z}_3$	-1	-1	0	-1	3
$\mathbf{z}_4$	-1	-1	-1	0	3

Additional snapshots (linearly dependent)					
	W1	W2	W3	W4	W5
$\mathbf{z}_5$	-1	-1	-1	-1	4
$\mathbf{z}_6$	-1	0	0	-1	2
$\mathbf{z}_7$	-1	-1	0	0	2
$\mathbf{z}_8$	-1	0	-1	0	2
$\mathbf{z}_9$	0	-1	-1	0	2
$\mathbf{z}_{10}$	0	-1	0	-1	2
$\mathbf{z}_{11}$	0	0	-1	-1	2
$\mathbf{z}_{12}$	-1	0	0	0	1
$\mathbf{z}_{13}$	0	-1	0	0	1
$\mathbf{z}_{14}$	0	0	-1	0	1
$\mathbf{z}_{15}$	0	0	0	-1	1

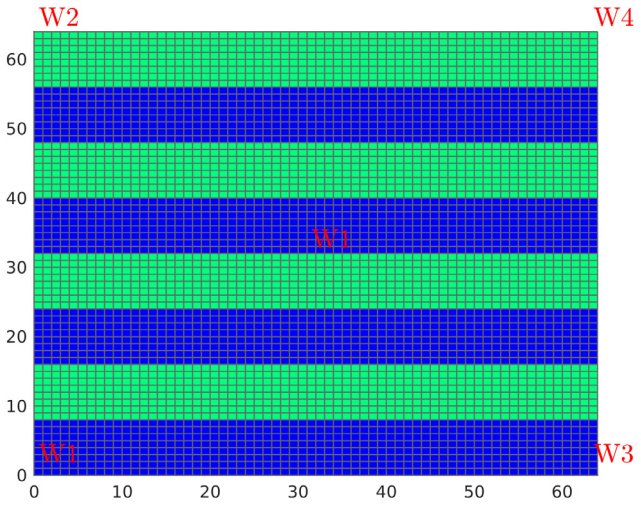

Fig. 1. Heterogeneous permeability, 5 wells.

Four wells are positioned in the corners with a bhp (bottom hole pressure) of −1 bar. One well is positioned in the center of the domain and has a bhp of +4 bars (see Fig. 1). Snapshots are obtained solving the system with different well configurations (see Table 1).

Table 2 shows the number of iterations required to reach convergence for the ICCG method and the deflation method with four linearly independent snapshots as deflation vectors DICCG$_4$, 15 linearly dependent snapshots DICCG$_{15}$ and the POD basis vectors, DICCG$_{POD_4}$.[2] For the deflation vectors of DICCG$_{POD_4}$ we plot the eigenvalues of the snapshot correlation matrix $\mathbf{R} = \frac{1}{15}\mathbf{X}^T\mathbf{X}$ (see Section 4) in Fig. 2. We observe that there are 4 eigenvalues much larger than the rest. These largest eigenvalues are responsible for the slow convergence of the ICCG method. For the DICCG$_{POD_4}$ method, we use the eigenvectors corresponding to the larger eigenvalues as deflation vectors.

In Table 2, for the ICCG method, we observe that the number of iterations increases if the contrast in the permeability increases. For the DICCG method with 4 linearly independent deflation vectors and 4 POD basis vectors, convergence is reached within one iteration and it does not change when we vary the contrast between permeability layers. However, for the case of 15 linearly dependent vectors, the solution is not reached within the 200 iterations, the maximum number of iterations allowed for this problem.

[2] The * means that the solution is not reached within the maximum number of iterations allowed for the problem.

Table 2
Number of iterations for different contrast in the permeability of the layers for the ICCG and DICCG methods.

σ_2 (mD)	10^{-1}	10^{-2}	10^{-3}
ICCG	90	115	131
$DICCG_4$	1	1	1
$DICCG_{15}$	200*	200*	200*
$DICCG_{POD_4}$	1	1	1

The * means that the solution is not reached within the maximum number of iterations allowed for the problem.

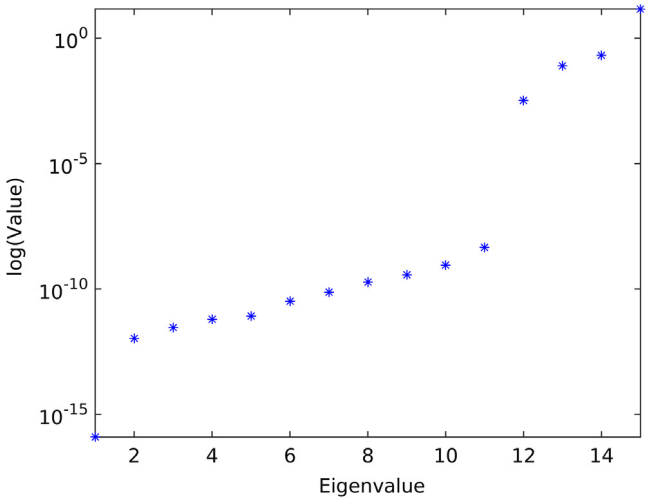

Fig. 2. Eigenvalues of the snapshot correlation matrix $\mathbf{R} = 1/15\mathbf{X}\mathbf{X}^T$, if 15 snapshots are used.

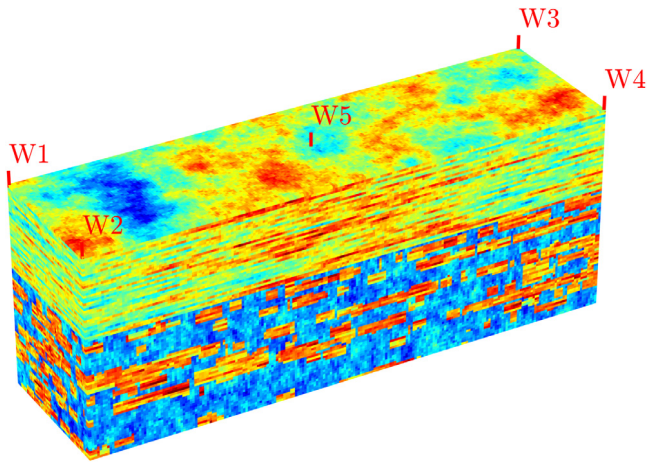

Fig. 3. SPE 10 benchmark, permeability field.

SPE 10 *model*

This model has large variations in the permeability coefficients, the contrast between coefficients is 3×10^7 [17]. The model contains $60 \times 220 \times 85$ cells (Fig. 3) and five wells, four of them located in the corners and one in the center of the domain.

Snapshots are obtained solving the system with different well configurations (see Table 1). As before, we simulate single-phase incompressible flow.

The number of iterations required to achieve convergence with the ICCG and DICCG methods for various grid sizes is presented in Table 3. In this table, we observe that for the ICCG method we require 1011 iterations to reach the desired

Table 3
Table with the number of iterations for ICCG and DICCG methods.

Method	Iterations
ICCG	1011
$DICCG_{15}$	2000*
$DICCG_4$	2
$DICCG_{POD_4}$	2

The * means that the solution is not reached within the maximum number of iterations allowed for the problem.

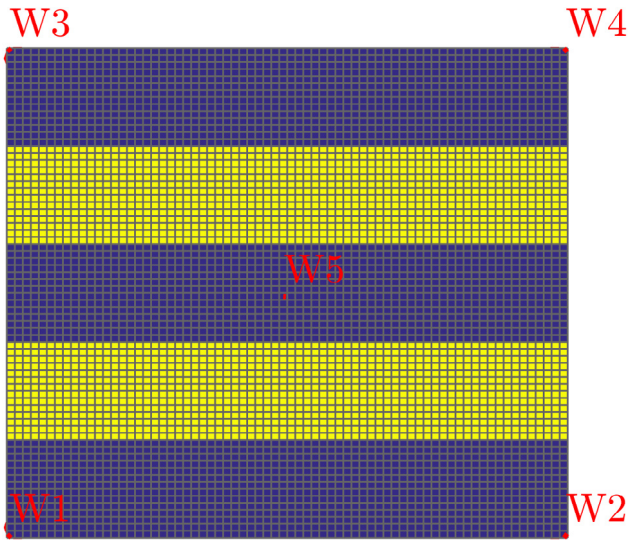

Fig. 4. Heterogeneous permeability, 5 wells, compressible problem.

accuracy. Meanwhile, for the deflated methods $DICCG_4$ and $DICCG_{POD_4}$ only two iterations are required. In theory, only one iteration is necessary to reach the solution with the deflated methods. However, the large contrast in the permeability field may require higher accuracy for the snapshots to find the solution with the deflation method, within one iteration, within the imposed tolerance [31,32]. In this case, the first iteration has a relative residual smaller than 10^{-10} for the $DICCG_4$ and $DICCG_{POD_4}$ methods, therefore, after a first iteration, we have already a good approximation.

For the deflated method with 15 linearly dependent snapshots ($DICCG_{15}$), we observe that the desired accuracy is not reached after 2000 iterations, as the deflation methods are linearly dependent, the deflation method is unstable because the matrix E is a nearly singular matrix.

6.3. Compressible problem

In this section we model single-phase flow through a porous medium for a case when the density depends on the pressure according to Eq. (4). We solve Eq. (12) for a fluid with a compressibility constant of $c_f = 1 \times 10^{-3}$. Eq. (12) is non-linear due to the dependence of the density on the pressure. Therefore, we need to linearize this equation via the Newton–Raphson (NR) method and to solve the resulting linear system. After linearization, we obtain the linear system (17) and we solve it with an iterative method; a summary of the procedure is presented in Algorithm 1. The simulation, with exception of the linear solvers, is performed with MRST. Automatic Differentiation (AD) is used for the NR loop [30]. The resulting linear system is solved with ICCG and DICCG methods. We compute the solution of the system for the first 10 time steps with the ICCG method. The rest of the time steps are solved with DICCG, using as deflation vectors the solution of the previous ten time steps and POD basis vectors computed from these solutions. The number of POD deflation vectors is specified for each problem.

We study an academic layered problem that consists of layers with two different permeability values (see Fig. 4). The first layer has a permeability of $\sigma_1 = 30$ mD, and the permeability of the second layer is varied; the permeability values of this second layer are $\sigma_2 = [3$ mD, 0.3 mD, 0.03 mD$]$. Therefore, the contrast between the layers is 10^1, 10^2 and 10^3. The domain is a square with five wells, four of which are positioned in the corners of the domain and one well in the center. The length of the domain is 70 m and the grid size is 35 grid cells in each dimension. We impose homogeneous Neumann boundary conditions on all boundaries.

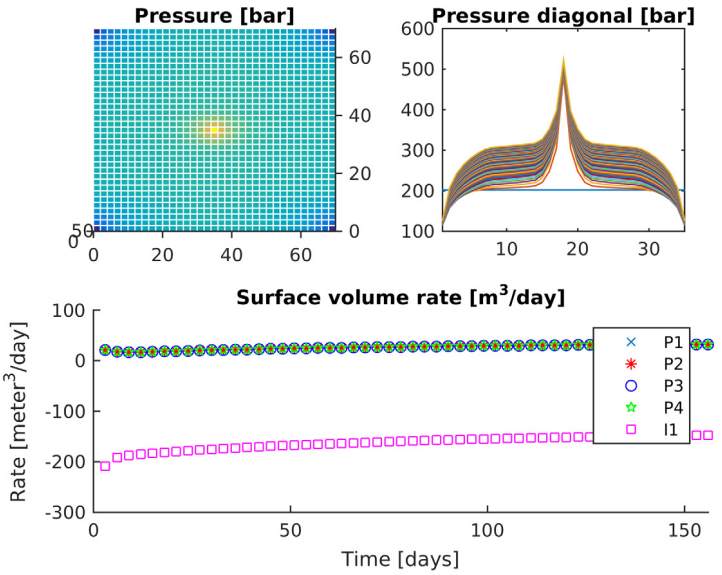

Fig. 5. Solution of the compressible problem solved with the ICCG method for a layered problem with a contrast between permeability layers of 10^1.

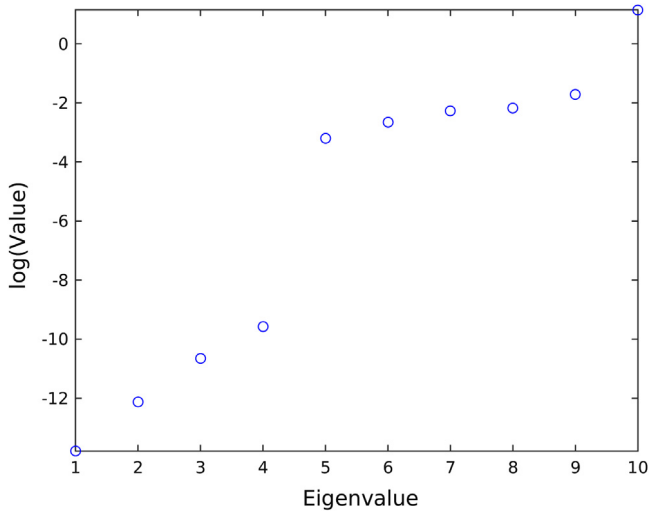

Fig. 6. Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \frac{1}{10}\mathbf{X}\mathbf{X}^T$, contrast between permeability layers of 10^1.

The initial pressure of the reservoir is set to 200 bars. The bottom hole pressure (bhp) in the corner wells is 100 bars and in the central well is 600 bars. The simulation was performed during 152 days with 52 time steps and a time step of 3 days. The tolerance of the NR method and the linear solvers is 10^{-5}.

In Fig. 5, the solution obtained with the ICCG method is presented for a contrast in permeability layers of 10^1. The upper left figure represents the pressure field at the final time step. The upper right figure represents the pressure across the diagonal joining the (1,1) and (35,35) grid cells for all the time steps. We observe the initial pressure (200 bars) across this diagonal and the evolution of the pressure field through time. In the lower figure, we observe the surface volume rate of the five wells during the simulation.

As mentioned before, for each time step, the previous 10 solutions are used as snapshots to compute the POD basis. The eigenvalues of the snapshot correlation matrix $\mathbf{R} = \frac{1}{10}\mathbf{X}\mathbf{X}^T$ constructed with the previous ten time steps are presented in Fig. 6 for the 20th time step, for a contrast between permeability layers of 10.

In Fig. 6, we observe that six eigenvalues are larger than the rest. Then, we use the eigenvectors corresponding to these six eigenvalues as deflation vectors (DICCG$_{POD_6}$) to solve this problem. For a contrast between permeability layers of 10^2 and 10^3 we have 7 larger eigenvalues, therefore, we use 7 POD basis vectors as deflation vectors (DICCG$_{POD_7}$). For all the

Table 4
Comparison between the ICCC and DICCG methods of the average number of linear iterations for the first NR iteration for various contrast between permeability layers.

1st NR iteration						
$\frac{\sigma_2}{\sigma_1}$	Total ICCG	Method	ICCG snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
10^1	780	$DICCG_{10}$	140	42	182	23
	780	$DICCG_{POD_6}$	140	84	224	29
10^2	624	$DICCG_{10}$	100	42	142	23
	624	$DICCG_{POD_7}$	100	42	142	23
10^3	364	$DICCG_{10}$	20	42	62	17
	364	$DICCG_{POD_7}$	20	42	62	17

Table 5
Comparison between the ICCC and DICCG methods of the average number of linear iterations for the second NR iteration for various contrast between permeability layers.

2nd NR iteration						
$\frac{\sigma_2}{\sigma_1}$	Total ICCG	Method	ICCG snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
10^1	988	$DICCG_{10}$	180	78	258	26
	988	$DICCG_{POD_6}$	180	198	378	38
10^2	832	$DICCG_{10}$	140	90	230	28
	832	$DICCG_{POD_7}$	140	154	294	33
10^3	884	$DICCG_{10}$	110	90	200	23
	884	$DICCG_{POD_7}$	110	150	260	29

experiments, only the first time step requires more than two NR iterations. Hence, we solely study the behavior of the linear solvers during the first two NR iterations.

In Tables 4 and 5 we compare the number of iterations necessary to reach convergence with the ICCG method and the deflation methods $DICCG_{10}$, $DICCG_{POD_6}$ and $DICCG_{POD_7}$.

For the first NR iteration (see Table 4), we observe a significant reduction in the total number of linear iterations. For the case when we have a contrast between permeability layers of 10^1, we observe that for the ICCG method, we need 780 linear iterations to compute the solution for the 52 time steps. By contrast, when we use the deflated method, we need 140 linear iterations to compute the snapshots during the first ten time steps (computed with ICCG) and 42 and 84 for the 42 remaining time steps computed with $DICCG_{10}$ and $DICCG_{POD_6}$. Then, we need in total 182 and 224 linear iterations to compute the solution for the 52 time steps, which is 23% and 29% of the linear iterations required with only the ICCG method.

When we have a contrast in permeability of 10^2, the required average of linear iterations to solve the 52 time steps is 624 for the ICCG method. With the deflated methods, taking into account the computation of the snapshots, we require 142 iterations for the $DICCG_{10}$ and $DICCG_{POD_7}$ methods, which is the 23% of the number of ICCG iterations. Finally, for a contrast between permeability layers of 10^3 we require 364 linear iterations for the 52 time steps with the ICCG method. Meanwhile, the required iterations for the $DICCG_{10}$ and $DICCG_{POD_7}$ methods is 62. That is 17% of the ICCG iterations (see Table 4).

For the second NR iteration (see Table 5), we also observe a significant reduction in the total number of linear iterations. For the case when we have a contrast between permeability layers of 10^1, with the $DICCG_{10}$ and $DICCG_{POD_6}$ methods, it is necessary to perform only 26% and 38% of the linear iterations required with ICCG.

When we have a contrast in permeability layers of 10^2, the required linear iterations are 28% and 33% of the ICCG iterations if we use the $DICCG_{10}$ and $DICCG_{POD_7}$ methods. For a contrast between permeability layers of 10^3, the $DICCG_{10}$ and $DICCG_{POD_7}$ methods require 23% and 29% of the number of ICCG iterations.

SPE 10 model

We study the complete SPE 10 benchmark, that consists of $60 \times 220 \times 85$ grid cells and has a contrast in permeability of 3×10^7. To solve the linear system obtained after the NR linearization, we use 10 snapshots (the previous 10 time step solutions), and POD basis vectors as deflation vectors. The simulation was performed during 152 days with 52 time steps and a time step of 3 days. In Fig. 7 the eigenvalues of the snapshot correlation matrix are presented. We observe that there are 4 eigenvalues larger than the rest, which implies that most of the information is contained in these eigenvalues. Therefore, we study the deflation method with 10 snapshots as deflation vectors and 4 POD basis vectors, the largest eigenvectors corresponding to the largest eigenvalues in Fig. 7.

For the first NR iteration, we observe that the average number of iterations required for the ICCG method is considerably reduced. For the ICCG method we require 10 173 iterations for the first NR iteration and 10 231 for the second (see Tables 6 and 7). With the deflated methods $DICCG_{10}$ and $DICCG_{POD_4}$, for the first NR iteration, we only need to perform 28% and 32% of the linear iterations required with the ICCG method.

For the second NR iteration, the deflated methods require only 20% of the ICCG linear iterations. We also observe, that for the first NR iteration, we need 1770 linear iterations to compute the ten initial snapshots (computed with ICCG) and 1134 to

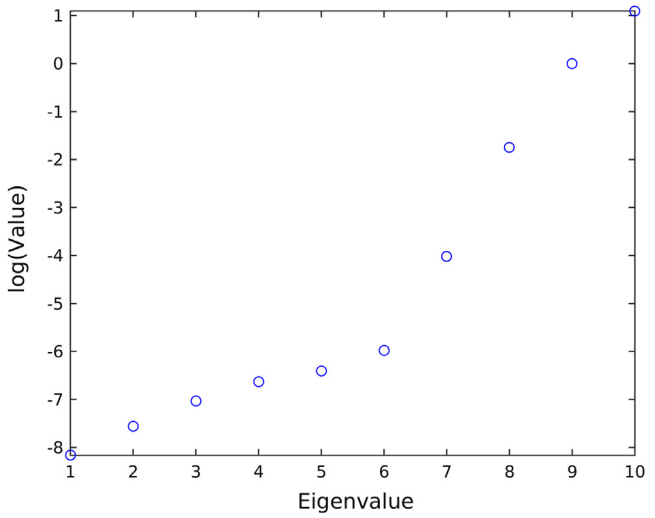

Fig. 7. Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \frac{1}{10}\mathbf{X}\mathbf{X}^T$, time step 20, full SPE 10 benchmark.

Table 6
Average number of linear iterations for the first NR iteration, full SPE 10 benchmark.

1st NR iteration					
Total ICCG	Method	ICCG snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
10 173	DICCG_{10}	1770	1134	2904	28
10 173	DICCG_{POD_4}	1770	1554	3324	32

Table 7
Average number of linear iterations for the second NR iteration, full SPE 10 benchmark.

2nd NR iteration					
Total ICCG	Method	ICCG snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
10 231	DICCG_{10}	1830	200	2030	20
10 231	DICCG_{POD_4}	1830	200	2030	20

compute the solution of the rest of the solutions (computed with DICCG). For the second NR iteration, the number of linear iterations is 1830 for the ten initial snapshots and 200 for the deflated methods. This shows that the largest amount of work is carried out by the computation of the snapshots obtained with the ICCG method, which is more evident for the second NR iteration.

7. Conclusions

In this work, we combine ICCG preconditioning with deflation and POD methods to accelerate the convergence of CG method for large systems and systems with high-contrast in permeability. The Deflated Conjugated Gradient preconditioned with Incomplete Cholesky method (DICCG) is studied with *snapshots*, solutions of the system with diverse characteristics, and POD basis vectors as deflation vectors. The number of iterations required with DICCG is compared with the number of iterations required with the ICCG method for the same problems. The stopping criteria used for ICCG and DICCG is the same. Therefore, the solution has the same accuracy for both methods.

Flow through a porous medium is studied for an incompressible and a compressible fluid. We study an academic layer problem with different permeability values in the layers and the complete SPE 10 benchmark problem (1,122,000 cells).

To solve the incompressible problem, we propose the use of solutions of the problem with different well configurations as deflation vectors. We observe that the number of linear iterations required with ICCG is reduced to only a few iterations when using DICCG, and this number is independent of the contrast in permeability layers for the deflation methods. Results also show that, if we have a linearly dependent set of deflation vectors, we have an unstable method that leads to a bad approximation of the solution. Combination of POD with deflation techniques is shown to be a way to obtain the main information about the system to speed-up the iterative method and to avoid instabilities.

For the compressible case, we propose the use of solutions of previous time steps, *snapshots*, and POD basis vectors computed from these snapshots as deflation vectors. With DICCG we reduce the number of iterations up to 20% of the number

Table A.8
Notation.

Symbol	Quantity	Unit
ϕ	Rock porosity	
$\mathbf{K}$	Rock permeability	Darcy (D)
c_r	Rock compressibility	Pa^{-1}
$\mathbf{v}$	Darcy's velocity	m/d
ρ	Fluid density	kg/m^3
μ	Fluid viscosity	Pa s
p	Pressure	Pa
g	Gravity	m/s^2
c_f	Fluid compressibility	Pa^{-1}
q	Sources	

of iterations of the ICCG method with only a small increase in the number of flops. For the problem with four deflation vectors, each DICCG iteration needs around 1.4 times the number of flops required with the ICCG iteration.

We observe that the performance of the DICCG method with snapshots and POD basis vectors as deflation vectors is similar. The required number of POD basis vectors to achieve a good acceleration of the method depends on the problem. However, only a limited number of POD vectors is necessary to obtain a good speed-up (seven at most for the problems here studied). For the SPE 10 problem, we reduce the number of deflation vectors to 4 POD basis vectors. For this number of deflation vectors, the computation of the POD basis requires around 10^4 flops, which is less than the number of cells of the problem.

The deflation techniques here presented are not restricted to these methods and could be combined with different preconditioners, e.g. SSOR or AMG, and diverse iterative methods.

Acknowledgments

We like to thank the ' Consejo Nacional de Ciencia y Tecnología (CONACYT)', the ' Secretaría de Energía (SENER)' and the Mexican Institute of Petroleum (IMP) which, through the programs: 'Formación de recursos humanos especializados para el sector hidrocarburos (CONACYT-SENER Hidrocarburos) (No. Convocatoria: 291128) and 'Programa de Captación de Talento, Reclutamiento, Evaluación y Selección de Recursos Humanos (PCTRES), have sponsored this work.

Appendix A. List of notation

In this appendix we present the notation for terms used in this work (see Table A.8).

Appendix B. Stopping criteria

When we use an iterative method, we always want that our approximation is close enough to the exact solution. In other words, we want that the error [28, page. 42]:

$$\|\mathbf{e}^k\|_2 = \|\mathbf{x} - \mathbf{x}^k\|_2,$$

or the relative error:

$$\frac{\|\mathbf{x} - \mathbf{x}^k\|_2}{\|\mathbf{x}\|_2},$$

is small.

When we want to chose a stopping criteria, we could think that the relative error is a good candidate, but it has the disadvantage that we need to know the exact solution to compute it. What we have instead is the residual

$$\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k,$$

that is actually computed in each iteration of the CG method. There is a relationship between the error and the residual that can help us with the choice of the stopping criteria.

$$\frac{\|\mathbf{x} - \mathbf{x}^k\|_2}{\|\mathbf{x}\|_2} \leq \kappa_2(A)\frac{\|\mathbf{r}^k\|_2}{\|\mathbf{b}\|_2}.$$

With this relationship in mind, we can choose the stopping criteria as an ϵ for which

$$\frac{\|\mathbf{r}^k\|_2}{\|\mathbf{b}\|_2} \leq \epsilon.$$

But we should keep to have in mind the condition number of the matrix **A**, because the relative error will be bounded by:

$$\frac{\|\mathbf{x}-\mathbf{x}^k\|_2}{\|\mathbf{x}\|_2} \leq \kappa_2(A)\epsilon.$$

Appendix C. Singular value decomposition for POD

If we perform SVD in **X**, we obtain the following matrices $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, $\mathbf{U} \in \mathbb{R}^{n\times n}$, $\mathbf{\Sigma} \in \mathbb{R}^{n\times m}$, $\mathbf{V} \in \mathbb{R}^{m\times m}$.
To obtain the eigenvectors of **X**, it is necessary to construct the matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T \in \mathbb{R}^{n\times n}$.
However, if the problem is large, the resulting matrix is large and the SVD can be expensive. Instead, we can compute the eigenvalues and eigenvectors from the much smaller matrix $\mathbf{R}^T = \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{m\times m}$. For this matrix, the SVD is:

$$\begin{aligned}\mathbf{R}^T &= \mathbf{X}^T\mathbf{X}\\ &= (\mathbf{U}\Sigma\mathbf{V}^T)^T\mathbf{U}\Sigma\mathbf{V}^T\\ &= \mathbf{V}\Sigma^T\mathbf{U}^T\mathbf{U}\Sigma\mathbf{V}^T,\ \mathbf{U}^T\mathbf{U} = \mathbf{I}\\ &= \mathbf{V}\Lambda^T\mathbf{V}^T,\ \Lambda^T = \Sigma^T\Sigma \in \mathbb{R}^{m\times m}.\end{aligned}$$

From where we obtain **V** and Λ^T. Then we can compute **U** as follows:

$$\mathbf{U} = \mathbf{X}\mathbf{V}(\Lambda^T)^{-\frac{T}{2}} = \mathbf{X}\mathbf{V}(\Lambda^T)^{\frac{1}{2}},$$

that are the left-singular values of **X**.

Appendix D. Deflation method

In this appendix, we explain how to obtain the solution of the linear system of Eq. (18) with deflation. Some properties of the matrices used for deflation that will help us to find the solution of system (18) are [19]:

(a) $\mathbf{P}^2 = \mathbf{P}$.
(b) $\mathbf{A}\mathbf{P}^T = \mathbf{P}\mathbf{A}$.
(c) $(\mathbf{I}-\mathbf{P}^T)\mathbf{x} = \mathbf{Q}\mathbf{b}$.
(d) $\mathbf{P}\mathbf{A}\mathbf{Q} = \mathbf{0}^{n\times n}$.
(e) $\mathbf{P}\mathbf{A}\mathbf{Z} = \mathbf{0}^{n\times l}$.

To obtain the solution of the linear system (18), we start with the splitting:

$$\mathbf{x} = \mathbf{x} - \mathbf{P}^T\mathbf{x} + \mathbf{P}^T\mathbf{x} = (\mathbf{I}-\mathbf{P}^T)\mathbf{x} + \mathbf{P}^T\mathbf{x}. \tag{D.1}$$

Multiplying expression (D.1) by **A**, using the properties of the deflation matrices, we have:

$$\begin{aligned}\mathbf{A}\mathbf{x} &= \mathbf{A}(\mathbf{I}-\mathbf{P}^T)\mathbf{x} + \mathbf{A}\mathbf{P}^T\mathbf{x}, && \textit{Property}:\\ \mathbf{A}\mathbf{x} &= \mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{A}\mathbf{P}^T\mathbf{x}, && (c)\\ \mathbf{b} &= \mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{P}\mathbf{A}\mathbf{x}, && (b),\end{aligned}$$

multiplying by **P** and using the properties $\mathbf{P}\mathbf{A}\mathbf{Q} = \mathbf{0}^{n\times n}$ and $\mathbf{P}^2 = \mathbf{P}$, properties (d) and (a), we have:

$$\begin{aligned}\mathbf{P}\mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{P}^2\mathbf{A}\mathbf{x} &= \mathbf{P}\mathbf{b},\\ \mathbf{P}\mathbf{A}\mathbf{x} &= \mathbf{P}\mathbf{b},\end{aligned}$$

where $\mathbf{P}\mathbf{A}\mathbf{x} = \mathbf{P}\mathbf{b}$ is the deflated system. Since **PA** is singular, the solution of Eq. (D.2) can contain components of the null space of **PA**, ($\mathcal{N}(\mathbf{PA})$). A solution of this system, called the deflated solution, is denoted by $\hat{\mathbf{x}}$. Then, the linear system to solve is:

$$\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{P}\mathbf{b}. \tag{D.2}$$

As the solution of Eq. (D.2) can contain components of $\mathcal{N}(\mathbf{PA})$, $\hat{\mathbf{x}}$ can be decomposed as:

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{y}, \tag{D.3}$$

with $\mathbf{y} \in \mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{PA})$, and **x** the solution of Eq. (18).
Note: If $\mathbf{y} \in \mathcal{R}(\mathbf{Z})$, then

$$\mathbf{y} = \sum_{i=1}^{m} \alpha_i \mathbf{z}_i,$$

$$\mathbf{PAy} = \mathbf{PA}(\mathbf{z}_1\alpha_1 + \cdots + \mathbf{z}_m\alpha_m) = \mathbf{PAZ}\alpha,$$

from property (e) we have:

$$\mathbf{PAy} = \mathbf{0}.$$

Therefore $\mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{PA})$, and using property (b) we have:

$$\mathbf{PAy} = \mathbf{AP}^T\mathbf{y} = \mathbf{0}.$$

As $\mathbf{A}$ is invertible, we have:

$$\mathbf{P}^T\mathbf{y} = \mathbf{0}. \tag{D.4}$$

Multiplying Eq. (D.3) by $\mathbf{P}^T$ we obtain:

$$\mathbf{P}^T\hat{\mathbf{x}} = \mathbf{P}^T\mathbf{x} + \mathbf{P}^T\mathbf{y}$$

substituting Eq. (D.4) we arrive to:

$$\mathbf{P}^T\hat{\mathbf{x}} = \mathbf{P}^T\mathbf{x}. \tag{D.5}$$

Substitution of Eq. (D.5) and property (c) in Eq. (D.1) leads to:

$$\mathbf{x} = \mathbf{Qb} + \mathbf{P}^T\hat{\mathbf{x}}, \tag{D.6}$$

which gives us the relation between $\hat{\mathbf{x}}$ and $\mathbf{x}$.

D.1. Stopping criterion for the deflation method

In Appendix B, we describe the stopping criterion used for the CG method, which is:

$$\frac{\|\mathbf{r}^k\|_2}{\|\mathbf{b}\|_2} \leq \epsilon,$$

where, $\mathbf{r}$ is the residual of the CG method. For the DCG method, the stopping criterion used depends on the residual of the deflated system, i.e.

$$\frac{\|\hat{\mathbf{r}}^k\|_2}{\|\mathbf{b}\|_2} \leq \epsilon \tag{D.7}$$

with $\hat{\mathbf{r}}$ the residual of the deflated system. This residual can be written as:

$$\begin{aligned}
\hat{\mathbf{r}}^k &= \mathbf{P}(\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}^k) \\
&= \mathbf{Pb} - \mathbf{AP}^T\hat{\mathbf{x}}^k && \text{see Appendix D, property (b)} \\
&= \mathbf{Pb} - \mathbf{AP}^T\mathbf{x}^k && \text{see Eq. (D.5)} \\
&= \mathbf{Pb} + \mathbf{AQb} - \mathbf{Ax}^k && \text{see Appendix D, property (c)} \\
&= \mathbf{b} - \mathbf{Ax}^k = \mathbf{r}^k. && \text{see Eq. (22)}
\end{aligned}$$

Therefore, the residual of the DCG method is the same as the residual of the CG method.

References

[1] C. Vuik, A. Segal, L. Yaakoubi, E. Dufour, A comparison of various deflation vectors applied to elliptic problems with discontinuous coefficients, Appl. Numer. Math. 41 (1) (2002) 219–233.
[2] M. Benzi, Preconditioning techniques for large linear systems: a survey, J. Comput. Phys. 182 (2) (2002) 418–477.
[3] A. Antoulas, Approximation of Large-Scale Dynamical Systems, SIAM, 2005.
[4] W. Schilders, H. Van der Vorst, J. Rommes, Model Order Reduction: Theory, Research Aspects and Applications, Vol. 13, Springer, 2008.
[5] A. Quarteroni, G. Rozza, Reduced Order Methods for Modeling and Computational Reduction, Vol. 9, Springer, 2014.
[6] T. Heijn, R. Markovinovic, J.D. Jansen, et al., Generation of low-order reservoir models using system-theoretical concepts, SPE J. 9 (02) (2004) 202–218.
[7] P.T.M. Vermeulen, A.W. Heemink, C.B.M. Te Stroet, Reduced models for linear groundwater flow models using empirical orthogonal functions, Adv. Water Resour. 27 (1) (2004) 57–69.
[8] R. Markovinović, J.D. Jansen, Accelerating iterative solution methods using reduced-order models as solution predictors, Internat. J. Numer. Methods Engrg. 68 (5) (2006) 525–541.
[9] J. van Doren, R. Markovinović, J.D. Jansen, Reduced-order optimal control of water flooding using proper orthogonal decomposition, Comput. Geosci. 10 (1) (2006) 137–158.
[10] M.A. Cardoso, L.J. Durlofsky, P. Sarma, Development and application of reduced-order modeling procedures for subsurface flow simulation, Internat. J. Numer. Methods Engrg. 77 (9) (2009) 1322–1350.

[11] P. Astrid, G. Papaioannou, J.C. Vink, J.D. Jansen, Pressure preconditioning using proper orthogonal decomposition, in: SPE Reservoir Simulation Symposium. Society of Petroleum Engineers, 2011 SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 2011, pp. 21–23.
[12] S. Krogstad, et al., A sparse basis POD for model reduction of multiphase compressible flow, in: SPE Reservoir Simulation Symposium, Society of Petroleum Engineers, 2011.
[13] Y. Efendiev, J. Galvis, E. Gildin, Local–global multiscale model reduction for flows in high-contrast heterogeneous media, J. Comput. Phys. 231 (24) (2012) 8100–8113.
[14] R. Jiang, Pressure Preconditioning Using Proper Orthogonal Decomposition, Master's thesis, Stanford University, 2013.
[15] M. Pasetto, M. Ferronato, M. Putti, A reduced order model-based preconditioner for the efficient solution of transient diffusion equations, Internat. J. Numer. Methods Engrg. (2016).
[16] C. Vuik, A. Segal, J.A. Meijerink, An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients, J. Comput. Phys. 152 (1999) 385.
[17] M.A. Christie, M.J. Blunt, Tenth SPE comparative solution project: a comparison of upscaling techniques, SPE Reservoir Eng. Eval. 4 (4) (2001) 308–317.
[18] H. Klie, M.F. Wheeler, K. Stueben, T. Clees, et al., Deflation AMG solvers for highly ill-conditioned reservoir simulation problems, in: SPE Reservoir Simulation Symposium, Society of Petroleum Engineers, 2007.
[19] J.M. Tang, R. Nabben, C. Vuik, Y. Erlangga, Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods, J. Sci. Comput. 39 (3) (2009) 340–370.
[20] K. Carlberg, V. Forstall, R. Tuminaro, Krylov-subspace recycling via the POD-augmented conjugate-gradient method, SIAM J. Matrix Anal. Appl. 37 (3) (2016) 1304–1336.
[21] K. Aziz, A. Settari, Petroleum Reservoir Simulation, Chapman & Hall, 1979.
[22] Z. Chen, G. Huan, Y. Ma, Computational Methods for Multiphase Flows in Porous Media, SIAM, 2006.
[23] J.D. Jansen, A Systems Description of Flow Through Porous Media, Springer, 2013.
[24] G. Golub, C.V. Loan, Matrix Computations., third ed., Johns Hopkins University Press, Baltimore, MD, USA, 1996.
[25] J. Tang, Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems, Ph.D. thesis, Delft University of Technology, 2008.
[26] M. Clemens, M. Wilke, R. Schuhmann, T. Weiland, Subspace projection extrapolation scheme for transient field simulations, IEEE Trans. Magn. 40 (2) (2004) 934–937.
[27] B. Smith, P. Bjorstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, New York, 1996.
[28] Y. Saad, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.
[29] G. Strang, Linear Algebra and Its Applications, Wellesley-Cambridge Press, 2009.
[30] K.A. Lie, An Introduction to Reservoir Simulation Using MATLAB: User guide for the Matlab Reservoir Simulation Toolbox (MRST), SINTEF ICT, 2013.
[31] G.B. Diaz Cortes, C. Vuik, J.D. Jansen, Physics-based pre-conditioners for large-scale subsurface flow simulation, Report 16-3, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, 2016.
[32] G.B. Diaz Cortes, C. Vuik, J.D. Jansen, Physics-based pre-conditioners for large-scale subsurface flow simulation, in: Proceedings of the 15th European Conference on the Mathematics of Oil Recovery, ECMOR XV, 2016.